*“Things would have changed if I had timely access to electronic medical records”*Regina Holliday

|  |  |
| --- | --- |
| *Concepts to cover*   * Overview of HealthVault APIs * Examples for Get and PutThings API * Interacting with HealthVault   + People - HealthVault account and records   + Devices – APIs / Connection Center   + Applications |  |

As a platform HealthVault provides an innovative access management and programming interfaces for applications and devices to access a user’s health information.

In the previous chapter we discovered how to fetch and manipulate data stored in HealthVault. This chapter takes a closer look at the application programming interface offered by HealthVault to enable this interaction in a programmatic fashion. We will discuss various ways in which an application or device can interface with the HealthVault platform. The code samples will use .NET interfaces because they fit well with HealthVault, but the same interfaces are available in Java, PHP & other languages. The chapter will introduce the elements of programming that give the programmer access to data in HealthVault. Towards the end, we will discuss various architectural options available for interface an application or device with HealthVault. We’ll start by discussing accounts, because the first task is to get access to your own account.

# Accounts and Records

HealthVault provides innovative access management to let a family health manager access and manage the records of various family members. Mom, serving as the family health manager, can create records for her husband and children. In Fig 3-1, Jane has created account for her husband, Tom and two kids, Chris and Sarah. She has full access to all information in her families HealthVault records.

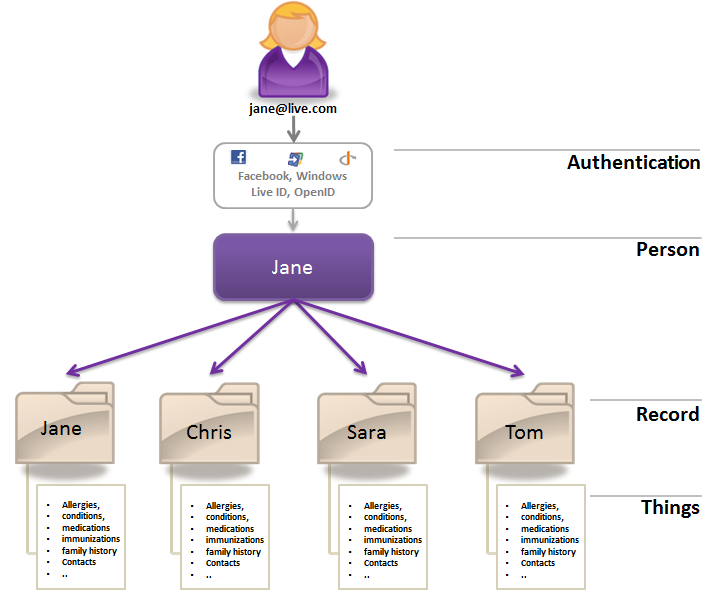
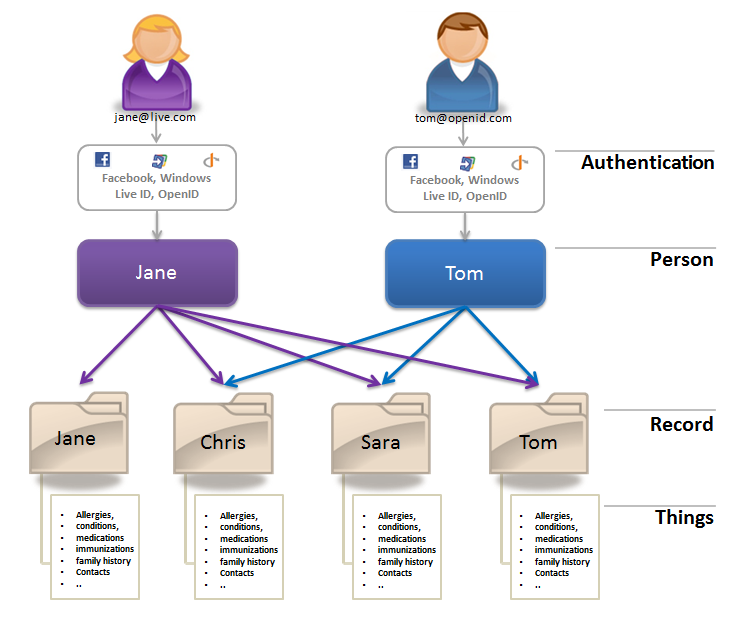


Fig 3-1. Multiple records under one HealthVault account

Additionally, HealthVault enables the same records to be accessed through multiple accounts. Full access can be thought of as custodial access to the record. In Fig 3-2, Jane has full access to health information of her family. Tom has also signed up to share the responsibility of managing health information of their kids Chris and Sarah, and also has full access to their health information.

  
Fig 3-2 Multiple accounts pointing to same HealthVault records

## Account Information

An application gets access to HealthVault account information through an API called Get-PersonInfo. This API returns a structure called PersonInfo, which in turn consists of the records associated with a HealthVault account.

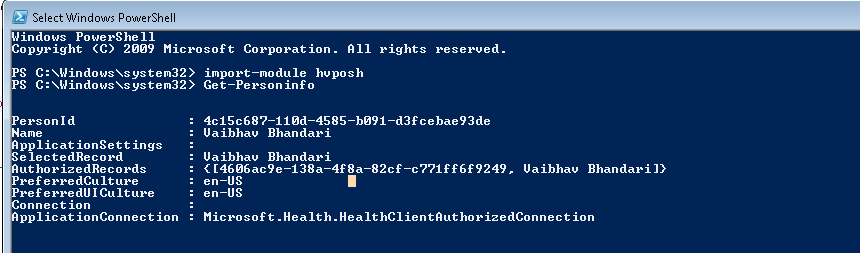


Fig 3-3 PersonInfo Data Structure

Using the HealthVault Powershell plug-in, you can try out the Get-PersonInfo API using the command get-personinfo. The structure returned consists of a unique PersonId for the account and a set of records identifier authorized to be used with the application for this particular account. Any pertinent information for the application is stored in ApplicationSettings. The user’s preferred language and display settings are stored in the PreferredCulture and PreferredUICulture fields respectively.

An application can decide to work with only one record at a time termed as *single record application* (SRA), or provide an interface to work with multiple records associated with the signed-in user termed as *multiple record application* (MRA). Section 2.4 describes how to enable each of these record management capabilities in detail.

# HealthVault Application Programming Interface

A HealthVault application interacts with two distinct resources:

<https://platform.healthvault.com/platform/wildcat.ashx>: the HealthVault Platform–

<https://account.healthvault.com>: the HealthVault Shell.

The HealthVault Platform provides XML over HTTP requests to manipulate data hosted by the service, and the HealthVault Shell provides account management, user authentication and other services.

HealthVault provides a development environment for partners to develop their applications. The development environment is hosted at <https://platform.healthvault-ppe.com/latform>, for HealthVault platform and <https://account.healthvault-ppe.com> for HealthVault Shell.

Each HealthVault application gets a unique identifier called AppID. Developers can get a free application identifier using the HealthVault Configuration Center (<https://config.healthvault-ppe.com>).

<aside> For our example application we are using an application ID that was already created for a HelloWorld sample application. In Chapter 4, we will create our own application. </aside>

## HealthVault Shell Interface

The HealthVault Shell provides its functionality primarily by redirecting the end-user’s browser. HealthVault Shell presents a secure user interface dialog in the browser. These dialogs help with user authentication, authorization, record selection, and managing the user’s experience around health data.

An application communicates its intention to HealthVault Shell using a URI construct like [https://account.healthvault.com/redirect.aspx?target={](https://account.healthvault.com/redirect.aspx?target=%7bShellTarget%7d&targetqs=%7bShellTargetParams%7d)**[ShellTarget](https://account.healthvault.com/redirect.aspx?target=%7bShellTarget%7d&targetqs=%7bShellTargetParams%7d)**[}&targetqs={ShellTargetParams}](https://account.healthvault.com/redirect.aspx?target=%7bShellTarget%7d&targetqs=%7bShellTargetParams%7d)

The **ShellTarget** parameter specifies the intent of the application, which could range from prompting the user to authorize the application to letting the user view their health items. Table 3-1 summarizes some of these targets; a detailed list of the Shell Targets is available on HealthVault MSDN at http://msdn.microsoft.com/en-us/library/ff803620.aspx.

|  |  |
| --- | --- |
| **HealthVault Shell Target** | **Purpose** |
| AUTH | Prompts the user to authorize himself and select record(s) to be used with an application |
| APPREDIRECT | Redirects a user to another HealthVault application. |
| CREATEACCOUNT | Allows an application to create a new HealthVault account and redirects to the application after account authorization |
| CREATEAPPLICATION | Enables a client or desktop application to create an instance of its application on the device. |
| RECONCILE | Enables an application to redirect to the HealthVault Shell for reconciling a CCR/CCD with a user record. |
| VIEWITEMS | Allows an application to redirect a user to view or create health record items using HealthVault Shell. |

Table 3-1 HealthVault Shell redirect Interface (partial list)

In certain circumstances, the HealthVault Shell needs to communicate with the application. For example, if a user wants to know the privacy statement of the application or if the user decides to not authorize permission for the application to access their health items, the HealthVault shell would then need to communicate with the application. HealthVault requires applications to register a “Redirect URL” for functionality they provide. The Redirect URL should be a secure (https) URL that can respond to request of this nature:

https://{application redirect url}?target={**ApplicationTarget**}&targetqs={ApplicationTargetParameters}

The **ApplicationTarget** specifies the desired action to get serviced; it could range from the user asking for a Privacy statement to the user rejecting the application’s authorization request. Table 3-2 summarizes some of these targets; a detailed list of the Application Targets is available on a HealthVault MSDN [page](http://msdn.microsoft.com/en-us/library/ff803620.aspx).

|  |  |
| --- | --- |
| **HealthVault Application Target** | **Purpose** |
| **APPAUTHSUCCESS** | Notifies the application that the user successfully logged in and/or granted authorization to the application. |
| **SIGNOUT** | Notifies the application that the user logged out of their HealthVault session. The application can then do cleanup and show a sign-out page. |
| **SELECTEDRECORDCHANGED** | Notifies the application that the user successfully change the selected record. Section 2.2 shows example of handling this. |
| PRIVACY | Notifies the application that the user wants to view its privacy statement. |
| SERVICEAGREEMENT | Notifies the application that the user wants to view its terms of use or service agreement. |

Table 3-2 HealthVault application targets (partial list)

## HealthVault Platform APIs

The HealthVault Platform provides a number of APIs to enable access to application and user data; these APIs are well documented [ <http://developer.healthvault.com/pages/methods/methods.aspx>]. The following discussion will focus on kinds of functionality provided by these APIs. Table 3-3 summarizes the APIs available from the HealthVault platform

|  |  |  |
| --- | --- | --- |
| **HealthVault API Category** | **API Names** | **Purpose** |
| Authentication | CreateAuthenticatedSessionToken,  RemoveApplicationRecordAuthorization,  NewApplicationCreationInfo,  NewSignupCode,  GetPersonInfo,  GetAuthorizedRecords | Authenticate an application and a user. |
| Reading Health Items | GetThings | A rich interface to retrieve Health items along with an associated digital signature or streamed BLOBs. |
| Adding & Updating Health Items | PutThings,  OverwriteThings  BeginPutBlob | Enable an application to add or update Health item data. |
| Delete Health Items | RemoveThings | Enables an application to delete data. |
| Patient Connect | AssociatePackageId,  BeginPutConnectPackageBlob,  CreateConnectPackage,  CreateConnectRequest,  GetAuthorizedConnectRequests,  DeletePendingConnectPackage,  DeletePendingConnectRequest | Enable clinical applications to create a temporary drop-off or permanent connection for consumers without having a web interface. |
| Asynchronous Processing | GetAuthorizedPeople,  GetUpdatedRecordsForApplication,  GetEventSubscriptions,  UpdateEventSubscription,  SubscribeToEvent,  UnsubscribeToEvent | Enable application to work asynchronously with HealthVault and create a publish/subscribe model. |
| Messaging | SendInsecureMessage,  SendInsecureMessageFromApplication | Enable applications to send messages to consumers using these APIs. |
| Terminology | GetVocabulary,  SearchVocabulary | Enable applications to retrieve or search terminologies hosted by HealthVault. |
| Application Management | SetApplicationSettings  GetApplicationSettings  AddApplication,  UpdateApplication | Enable an application to store a record specific setting, and manage derivative applications. |
| Service Discovery | GetServiceDefinition,  GetThingType | Help with service discovery. |
| OpenQuery | SaveOpenQuery,  GetOpenQueryInfo,  DeleteOpenQuery | These are hardly used, but they give the ability to run pre-canned queries for a health record. |

Table 3-3 HealthVault API Summary

### Authentication & Authorization APIs

Applications authenticate themselves to the HealthVault platform using the CreateAuthenticatedSessionToken API. Users are authenticated through HealthVault.com and applications can get tokens for authorization using the HealthVault Shell redirect interface [<http://msdn.microsoft.com/en-us/library/ff803620.aspx>].

<aside>CreateAuthenticatedSessionToken or CAST is the most commonly used HealthVault API. The API provides authentication token for clients as well as web applications. Most HealthVault wrappers provide an API for this purpose.</aside>

Individual methods are available for an application to fetch record and person authorization details. Most notably, NewApplicationCreationInfo is used by mobile clients to receive security keys from the HealthVault platform.

### Reading Health Items

The core function that lets an application read items from a user’s HealthVault record is GetThings. We will discuss this API in detail in section [link], but to summarize, this API provides the ability to query HealthVault, fetch a health item with granular details, and fetch large BLOB items such as images.

<aside>After CreateAuthenticatedSessionToken, GetThings is the most commonly used HealthVault API. </aside>

### Creating and Updating Health Items

The counterpart of GetThings is PutThings. It is used by most applications to update and create health items. We will discuss this function in detail in section 2.3.1.

OverwriteThings allows applications to force overwrites on existing health items. This API is not generally used.

HealthVault, unlike most personal health platforms, provides a mechanism to store large files such as medical images. Application can upload large chunks of information by using the BeginPutBlob API. It is fairly tricky to use, but there is good documentation on how to use it via raw XML interfaces at <http://msdn.microsoft.com/en-us/library/ff803584.aspx>, as well as use the HealthVault .NET SDK at <http://msdn.microsoft.com/en-us/library/ff803576.aspx>.

### Delete Health Items

DeleteThings is the one of the simplest HealthVault functions, allowing applications to delete individual health items from a user’s record. HealthVault keeps an audit trail of all operations, including the delete operation.

<aside> Only users can view the audit trail for Health items by using HealthVault Shell’s history functionality. Applications do not have access to the audit trail of a health item. </aside>

### Patient Connect

As discussed later in section [link], several clinical applications use HealthVault to send information to consumers either a single time or continually through back-end systems. CreateConnectPackage allows applications to create a one-time package for the user to receive in their HealthVault account, and DeleteConnectPackage allows applications to perform cleanup as necessary. On the other hand, CreateConnectRequest allows application to establish a continual link with a patient’s HealthVault record. Application can get the details needed to make the link by using GetAuthorizedConnectRequests.

<aside>HealthVault will delete the validated connect requests after a period of time. It is advised that applications calls GetAuthorizedConnectRequests daily or weekly to ensure that all validated connect requests are retrieved. </aside>

### Asynchronous Processing

HealthVault provides several mechanisms for an application to perform asynchronous processing. GetAuthorizedPeople gets information about the people that are authorized to use the application. This function paginates results using a PersonID-cursor and provides a way to query authorizations created after a given point in time. Applications have found this function useful to send email updates and reminders to their subscribers. Similarly, GetUpdatedRecordsForApplication retrieves a list of records for an application with things that have been updated since a specified date.

HealthVault provides a powerful publish/subscribe mechanism. Applications can subscribe to events around create, read, update, and delete operations on HealthVault thing-types. These events are registered with the platform using the SubscribeToEvent method. In addition to defining the subscribe event, the application registers a secure URI to which the HealthVault platform publishes events. The HealthVault Eventing mechanism is documented in detail with appropriate examples at | http://msdn.microsoft.com/en-us/library/gg681193.aspx.

The InstantPHR, <http://www.getrealconsulting.com/instantphr/>, application from GetReal Consulting is a good example of an application that uses the HealthVault asynchronous processing and eventing in particular to notify users of changes in their Health records.

### Messaging

Applications using HealthVault can send e-mail messages to HealthVault users. SendInsecureMessageFromApplication allows the application to choose the sender address and specify its domain.

<aside>The HealthVault Messaging APIs are insecure. It would be better for an institution to set up the Direct email protocol and send secure email to the HealthVault user. HealthVault users get free Direct email addresses.</aside>

### Terminology

Terminologies, also known as vocabularies, are a list of codes associated with well-known terms in a particular domain.] HealthVault hosts numerous terminologies. Most as tagged as wc, and are created by Microsoft. However several third party terminologies from the National Library of Medicine, USDA, HL7, and other institutions are also hosted.

<aside>  
You can use the PowerShell HealthVault plugin (Listing 2.4) to verify that approximately 150 terminologies are hosted by HealthVault.

**PS C:\> (Get-Vocabulary).Count  
150**Listing 2.4 Using HvPosh to use Get-Vocabulary  
</aside>

The terminologies can be accessed using the GetVocabulary API. Accessing the terminologies does not require user authentication; these are application only APIs. Some terminologies hosted by HealthVault, like RxNorm, are huge. RxNorm is a terminology that attempts to normalize all medication names and contains more than 200,000 entries. The SearchVocabulary API provides an XML interface as well as a JSON interface to search vocabularies. In fact, one can get an auto completion text box for entering a medication by using SearchVocabulary on RxNorm. The HealthVault user interface provides auto completion for medications, conditions, and other health item types using the SearchVocabulary API.

### Application Management

The SetApplicationSettings and GetApplicationSettings functions provide a way for applications to store and retrieve their record-specific settings in HealthVault. Information like theme selection by a particular user or order of authorized records can be stored in application settings. The multiple record management (MRA) functionality detailed in Section 2.4.2 can be implemented using these APIs.

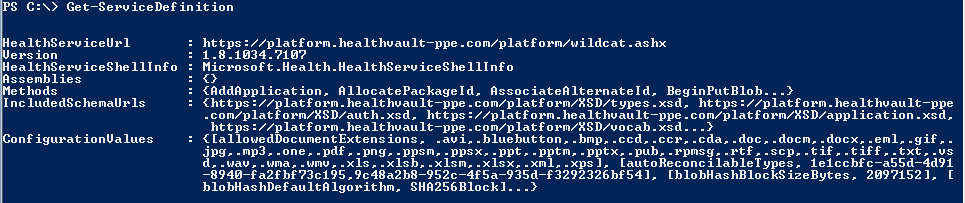
HealthVault also allows certain kind of applications, Master Applications, to create and manage other applications. Master Applications use the AddApplicaiton and UpdateApplication functions to manage the “child” applications they create.

<aside> There are very few Master Applications in the HealthVault ecosystem. The GoLive or publication bar for these applications is high. Once an application is created, it cannot be deleted. </aside>

### Service Discovery

The GetServiceDefinition function provides access to all the details of HealthVault applications, including Methods, Schemas, and Configurations for the service. Using GetServiceDefinition, an application can programmatically discover HealthVault service information and keep it up to date.

<aside>  
You can use the PowerShell HealthVault plugin (Fig 3-4) to explore GerServiceDefinition.

Figure 3-4 Using GetServiceDefinition in HvPosh  
</aside>

### Open Query

Open Query is an insecure mechanism for running preconfigured queries invoked with an identifier on HealthVault data. For example, the following query:

<https://platform.healthvault.com/platform/openquery.ashx?id=9C4C77CF-1DF0-4c41-BD3D-EC9232B5BC8A>

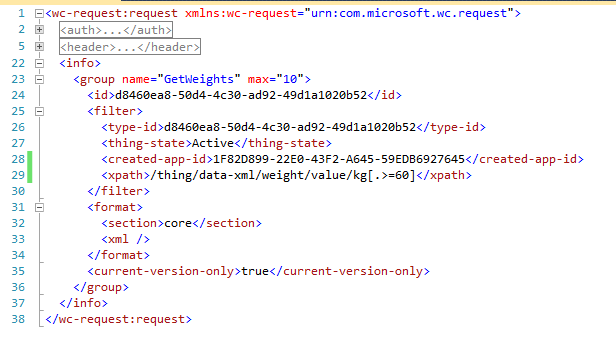
invokes a saved request that corresponds to the specified identifier. Only queries associated with GetThings can be saved with SaveOpenQuery. The invocation of the open query doesn’t require authentication and authorization and the HealthVault team discourages its use and it might be removed in future updates.

## Read and Write API - Diving Deep

The most important HealthVault functions an application should become familiar with are GetThings and PutThings. In the following sections we will dive deeper in to each of these functions, discuss how their treatment in the HealthVault .NET SDK, and see some examples. The code in this chapter can be a starting point for the more complex applications introduced in the rest of the book.

### GetThings

The best way to understand GetThings API is to look at the XML that an application would send to HealthVault platform to request a set of things.



Listing 3-1 GetThings XML request

Each GetThings request can have multiple queries called a **group** (Line 23). Each group is identified by its **name**. The response from HealthVault combines the items returned by the group; group-name is used to index items returned for a particular query group. The group element can take one or more attributes to control the results. The attribute used in the previous example is **max,** which tells the HealthVault platform to return the top 10 items for this query.

Each query group can contain <filter> sub elements to return particular items with specific identification. Listing 3-1 requests items with a specific type-id (Line 24). The **d84..52** identifier is associated with a particular instance of the Weight type. Multiple IDs can be specified in each request. Other elements, such as client identifiers and thing keys, can also be used instead of an ID. The **filter** also restricts results to those that have an active thing-state (Line 27) and were created using the Withings Application (Line 28). Items can also be filtered by using XPath; on Line 29 we are looking for Weight item, whose values are greater than 60kg, maybe because we know that the scale was misconfigured during this time. To sum up, the query in Fig 2.11 returns the core xml sections of the top ten weight elements that were created by the Withings application and have values greater than 60kg.

The format and quantity of information returned by the GetThings query can be controlled by **format** specifiers. Using the **section** tag, we can specify that we just want the **core** elements of the requested thing-types (Line 32). Other section tags could specify digital signatures, audit information, or *effective* permissions for the request. Using the **xml tag** (Line 33) one can run an XSL transform on the thing-types or choose from existing transforms available for the type. Our xml tag is empty, so no transform is run in the sample query. We will discuss MTT, STT, and other transforms in Chapter 4.

An application can request only the current-versions of HealthVault thing types (Line 35), although the platform does store older versions of the data.

To complete the discussion on the format of the GetThings request, note that I have collapsed the auth (Line2) and header (Line 5) tags. These elements specify the authorization information for the method and various other header elements, like method name, final-xsl, version, user, and application tokens.

Table 3-4 summarizes the querying ability of GetThings method. To learn more, please refer to the methods schema documentation at <http://developer.healthvault.com/pages/methods/methods.aspx> and associated HealthVault SDK reference at <http://msdn.microsoft.com/en-us/library/hh672196.aspx> .

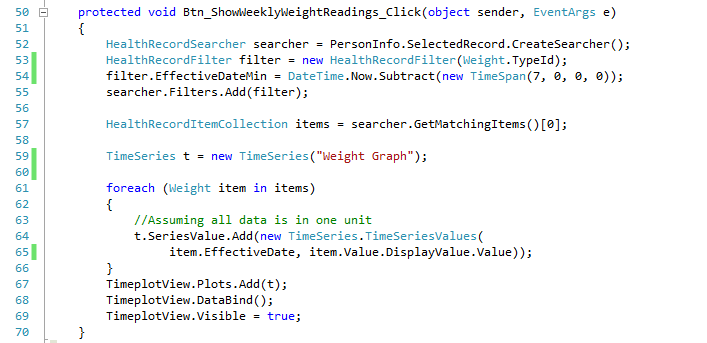
|  |  |  |  |
| --- | --- | --- | --- |
| **Search Criteria** |  | | |
| **.NET SDK Name XML Element Name Description** | | |
| **Group Attribute** | Name | name | identifies the group |
| Max | max | Maximum number of things to be returned |
| Max-Full | max-full | Maximum number of full things |
| **Identifiers** | ClientItemIds | client-thing-id | Client Id of the thing |
| ItemIds, | Id | Thing Instance Id |
| ItemKeys | key | Thing Key |
| **Filters** | EffectiveDateMax, EffectiveDateMin, UpdatedDateMax, UpdatedDateMin  CreatedDateMax, CreatedDateMin  CreatedPerson,  UpdatedPerson,  CreatedApplication, UpdatedApplication  XPath  States | eff-date-min  eff-date-max  updated-date-min  updated-date-max  created-date-min  created-date-max  created-person-id  updated-person-id  created-app-id  updated-app-id  xpath  thing-state | Various thing filters |
| **Formats** | **View** | Section | Section to be retrieved (core, audits, effectivepermissions, digitalsignatures) |
| Xml | Name of transform to apply |
| type-version-format | Version ID of type format |
| blob-payload-request | Sequence of blob-filters (blob names) and blob-format-spec (information, inline or streamed) |
| **Versions** | CurrentVersionOnly | current-version-only |  |

Table 3-4 GetThings query parameters

Having understood the paradigm through which one can access things from HealthVault using the GetThings methods, let’s look at how we can utilize querying to display Weight values in our application.

Our application currently fetches all the weight readings. However we want to be able to explore the readings on a graph one week at a time. Listing 3-2 shows how we can do it. As Line 52 shows, we begin by creating a searcher object for the record we are working with, and then create a new filter for Weight type (Line 53). Once we have the filter, we add properties to filter the value so that we get only weight items that are dated for previous week using the EffectiveDateMin (Line 54).

Once the query is constructed, we issue a GetThings request to HealthVault (Line 57). Since we have only one group, we index for the results in the first set of matching results (**GetMatchingItems()[0]**).



Listing 3-2 Using the EffectiveDateMin filter to get weekly data

After getting the matching items, we create a new TimeSeries (Line 59) and add each item individually to the series. The TimeSeries.ascx.cs file contains an implementation of this class. You can choose to use your own implementation with any other graphing library. In this example, I’m using the Flot javascript graphing library (<http://code.google.com/p/flot/>).TimeplotView (Line 67) is an instance of a user control that has a Flot graphing object.

In line 67-69 we add the constructed TimeSeries to the Graphing object and make it visible on the screen. Fig 3-5 shows results of running this information on the selected record.

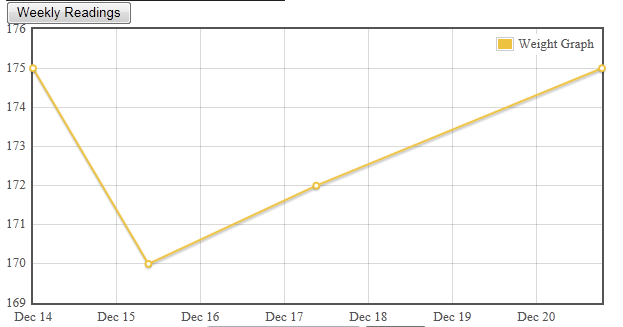
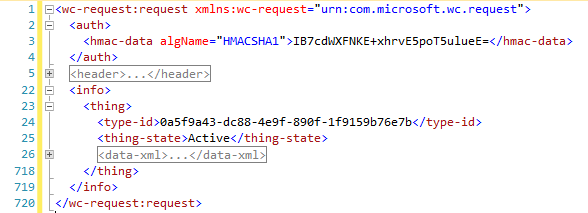


Fig 3-5 Filtering weight data for the week

### PutThings

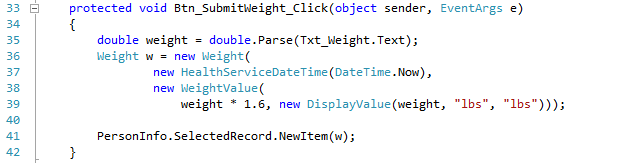
As with GetThings, we’ll begin our coverage of the PutThings API by looking at the XML an application would send to HealthVault to create or update a set of things. Compared to GetThings, this is a simple request.

  
Listing 3-3. Put-Things XML Sample

Each PutThings request can add instances of a **thing** (Line 23). Each thing has a type-id (Line 24) that identifies what kind of data item it is. In Listing 3-3, the thing is of type **Weight**. Because we are adding a weight thing, the **data-xml** (Line 23) part of this request needs to adhere to the schema for this particular type. In Chapter 4, we will discuss the thing-type schema. The important aspect of using this schema is to make sure to use a unique thing-id. In case of an update, the thing-id is the instance of weight you want to update. Additionally the thing-version-id element should be the version id of the element that is currently in the HealthVault. HealthVault offers optimistic concurrency: if an application tries to update an old thing that has already been updated by some other application, its version ID will have changed and the new put won’t succeed. The thing-version-id is critical to make sure one update does not override another.

Let’s modify our application to add a new Weight element to HealthVault. On the Default.aspx page we construct a text box to enter weight in pounds and associate a “save” button with it, with the action as “Btn\_SubmitWeight\_Click”.

Listing 3-4 illustrates how we will go about saving a new element to HealthVault. Notice the HealthVault-specific DateTime field (HealthServiceDateTime) on Line 37 and a way to differentiate the actual WeightValue from DisplayValue on Lines 38-39. HealthVault enables health items to have a flexible date time. It also stores v measurements in a canonical format and allows the user to see them in the format in which they entered the data. In the case of Weight, it is stored canonically in kilograms, but we assume the user prefers to enter and display the weight in pounds. So on Line 39 we multiply the value entered by the user by 1.6. The DisplayValue of the weight is in pounds (lbs), which the applications working with this type can use to show the value to the user



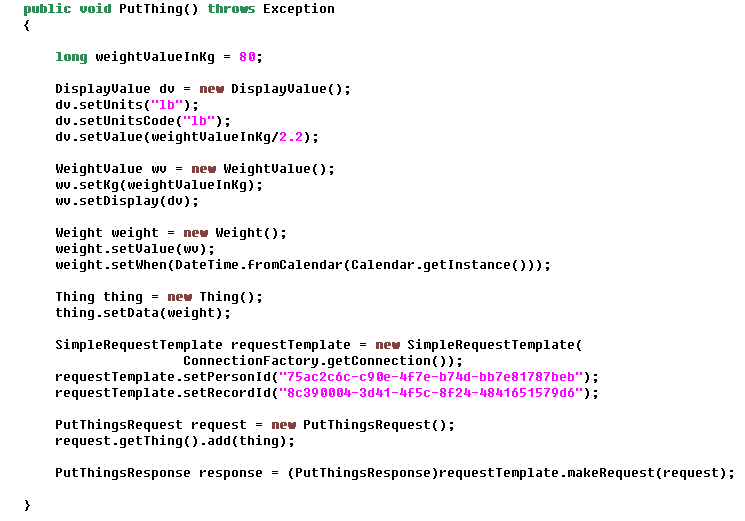
Listing 3-4 Put-Things Example

Having a well-formed weight health item, we can send it to HealthVault by calling NewItem method in HealthVault .NET SDK. Under the hood NewItem call issues a PutThings request to HealthVault platform. The HealthVault SDK also has as an UpdateItem method which saves changes to an existing item using the PutThings method.

To update an existing item, one essentially does the same thing as shown, except you use the UpdateItem SDK method instead of NewItem.

<aside>

The following code in Java is equivalent of the .NET Listing in 3-4. The OnlineRequestTemplate is generated using the libraries SimpleRequestTemplate, and it sets appropriate UserAuthToken and PersonId on the request.



Listing 3-5 Reading Weight Using Java SDK

</aside>

## Record Management – Diving deep

### Single Record Application (SRA)

For an application working with single record, HealthVault provides a simple mechanism to switch to another record using a switch record hyperlink. In order for an application to switch to a different record, the application needs to tell the HealthVault shell to give the user the ability to switch the record and have a pre-configured receiving end-point for HealthVault shell to send the user back to a notification to change the selected record, termed a SELECTEDRECORD. The application redirects the user with the hyperlink URL as in the following example, the AUTH target implies an authentication request, the *appid* implies the identifier associated with the calling application and the forceappauth string makes sure that the user gets an ability to change the record.

https://account.healthvault-ppe.com/redirect.aspx?target=**AUTH**&targetqs=appid%3D82d47a5a-d435-4246-895a-746c475090d3%26**forceappauth**%3Dtrue.

Using the HealthVault .NET Web SDK this can be done with following line of code. The last variable enables applications to pass any optional parameters which need to pass-through the url redirection.

this.RedirectToShellUrl("AUTH", "appid=" + this.ApplicationId.ToString() + "&forceappauth=true", "passthroughParam=optional");

As a simple example, Fig 2.4 shows how one can associate “switch account” functionality to an existing HealthVault application. Enabling this functionality involves two steps. First is to create an appropriate URL to send the user to HealthVault, Listing 3-6 shows the associated code. Second step in this process is to create a receiving URL so that HealthVault can send the user back to your application, Listing 3-7 shows the associated code required to configure a SelectedRecordChanged end-point page in the HealthVault SDK’s web.config file. These two steps are relatively simple and can be accomplished with any programming language, section 2.1 explains the HealthVault shell interface and section 3 lists the available libraries.

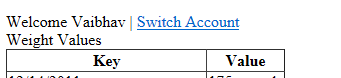
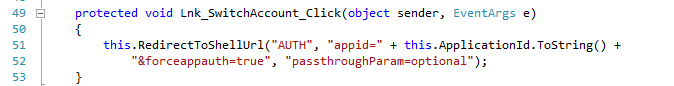


Fig 2.4. Adding ability to switch accounts.

In Listing 3-6 notice the Line 51, we are using the RedirectToShellUrl HealthVault SDK functionality to enable creating the appropriate redirection URL for HealthVault.



Listing 3-6 Adding Redirect Url in Default.aspx.cs

In Listing 3-7, we are creating a key in web.config file to associate a receiving end-point for selected record changed action. In the next section, we explain the HealthVault shell interface in detail.



Listing 3-7 Handling SelectRecordChanged Target in Web.Config

### Multiple Record Application (MRA)

The Mayo Clinic health manager works with all the records associated with the account and it termed as a multiple record application (MRA). As illustrated in Fig 3-6, I can switch from my record to my Mom’s record seamlessly within the Mayo Clinic application. This flexibility can be achieved in an application by including IsMRA=true while communicating with HealthVault Shell’s Authentication (AUTH) mechanism, as detailed in Section 2.1. Using this capability the application can then authenticate multiple records at the same time, and make requests to access health information for each one of them. For the application to remember the previous active record before the switch any associated record, it should store the current record identifier in the application settings associated with the person before making the switch.



Fig 3-6. An application working with multiple records

# HealthVault SDK and Open Source Libraries

The HealthVault team offers a .NET SDK [link]. Additionally, a number of open-source libraries offer higher-level abstractions for interacting with the HealthVault platform. This section outlines the level of abstractions available in each of these.

## HealthVault .NET SDK

The HealthVault .NET SDK is the official SDK available from Microsoft for working with the HealthVault platform [link]. The HealthVault team maintains this SDK and provides interfaces for all HealthVault interfaces.

This SDK does not support the HealthVault Client APIs for mobile phones, but it does support the HealthVault Client APIs for Windows Applications. The Shell Redirect Interface is supported, but not all capabilities are supported. Notably, this is the only SDK that supports signing health items and streaming large files to HealthVault. HealthVault uses Azure, Microsoft’s cloud storage service, to store these large files.

Throughout this book, we will be looking at code that uses this SDK, and refer to it as the HealthVault .NET SDK. Officially each major release of this SDK is supported for two years, and the SDK is currently compatible with .NET framework version 2.0.

The source code of this SDK is available for reference, but the license terms don’t allow modifications to the SDK.

## HealthVault Open Source Java SDK

This is the second most popular SDK for the HealthVault platform. The HealthVault open source Java SDK is available under a very permissive open source license [link]. This SDK was developed by members of HealthVault team and provides interfaces for most HealthVault interfaces. The source code of the SDK is available under the Microsoft Public license, and modifications and redistribution of this code are permitted for commercial and non-commercials purposes.

Notably, this SDK supports the HealthVault Client APIs for Android mobile phones, and provides a complete abstraction layer for Shell redirect interfaces. But it does not support Patient Connect, asynchronous processes, signing of health items, or streaming large files to HealthVault. However, there are samples or documentation available for signing [link] and streaming [link].

Additionally the SDK provides an object wrapper for thing-types using code generation tools. If these classes don’t meet your needs, you can use the schemas [link] and create suitable wrappers.

This SDK is fully available for JDK 1.6, however raw authentication is supported for JDK 1.4. The SDK is community supported, and patches for bug fixes or missing functionality are welcomed.

<aside>

Certificate Management

In case of the .NET SDK and Windows platform the HealthVault SDK offers the Application Manager tool to make it easy to work with public and private key for your application. In case of Java, the best way to handle certificates for the application is to create one using the keytool for the Java Development kit.

The following keytool command creates a public and private key pair for your application in the java keystore:

keytool -genkeypair -keyalg RSA -keysize 2048 -keystore keystore -alias java-wildcat -validity 9999

Note the algorithm used is RSA and keysize is 2 kilobytes, its recommend to have the keysize as large as your installation supports. The generated key pair is valid for 9999 days, and you can choose to configure it. The name of the key-pair is java-wildcat this needs to be added to hv-application.properties file in the java sdk.

The HealthVault platform needs to have the certificate associated with the public key of your application. The keytool can also be used to export this certificate. Following is a sample command to do the same:

keytool -export -alias java-wildcat -keystore keystore > my-pub.cer

java-wildcat is the name of the application’s key-pair and its exported as my-pub.cer.

</aside>

## HealthVault Open Source iOS Mobile Library

The HealthVault team provides an open source and community supported library for the iOS platform available at <https://github.com/microsoft-hsg/HealthVault-Mobile-iOS-Library> . This library provides basic functionality to authenticate mobile clients. It doesn’t provide support for any additional HealthVault features.

Applications such as iTriage, <http://www.itriagehealth.com/>,have used this library to create HealthVault iOS applications.

This library is available under the Apache 2.0 open source license, and modifications and redistribution of this code are permitted for commercial and non-commercials purposes.

## HealthVault OpenSource Windows Phone Library

Like the iOS library, the Windows Phone library [link] provides an authentication abstraction for Windows Phone mobile clients. Applications such livescape( <http://livescape.mobi/>) have used this library to create HealthVault-enabled Windows Phone applications.

This library is available under the Apache 2.0 open source license, and modifications and redistribution of this code are permitted for commercial and non-commercials purposes.

In Chapter 5, we will walk through a detailed application showing how to work with HealthVault mobile interfaces.

## HealthVault Open Source Python, PHP and Ruby Library

The HealthVault team has helped create Python, PHP , and Ruby libraries. These libraries are primarily driven by partners and provide the basic authentication layer for working with the HealthVault service. Applications such as TrailX (Python), Teladoc (PHP) and podfitness (Ruby) have used these libraries to create successful HealthVault applications.

These libraries are available under the Apache 2.0 open source license, and modifications and redistribution of this code are permitted for commercial and non-commercial purposes.

Table 2.2 summarizes the functionality available in various HealthVault libraries.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SDK library** | **Distribution** | **Supported Platform** | **Features Available** | **License & Support** |
| HealthVault .NET | MSDN | Windows XP, Vista, 7 (.NET 2.0) | All HealthVault features | MS-RL  Microsoft Supported |
| Java | Codeplex | JDK 1.6  JDK 1.4 (limited) | Authentication, Method wrappers, Thing-Type Wrappers. | MS-PL  Community Support |
| Java | Codeplex | Android (1.6+) | Authentication, Thing-type wrappers | MS-PL  Community Support |
| iOS | GitHub | iOS 4.0+ | Mobile Authentication | Apache 2.0  Community Support |
| Windows Phone | Codeplex | Windows Phone 7+ | Mobile Authentication | Apache 2.0  Community Support |
| Python | Google Code | Python 2.7 | Authentication (Basic) | Apache 2.0  Community Support |
| PHP | SourceForge | PHP | Authentication (Basic) | Apache 2.0  Community Support |
| Ruby | RubyForge | Ruby | Authentication (Basic) | Apache 2.0  Community Support |

Table 3-5 HealthVault SDK and Open Source Libraries

# Interfacing with HealthVault

We touched on the HealthVault APIs and interface; these interfaces are usually combined in multiple ways to create integration architectures with HealthVault. This section discusses high-level options for integrating applications and devices with HealthVault. This discussion should be useful for understanding different architectural patterns available for interfacing devices and applications with HealthVault.

## Device Connectivity

As of this writing, more than 80 types of devices connect with HealthVault. These devices vary from pedometers and weighing scales to blood pressure meters and pulse oximeters. Fig 3-7 shows the various interfaces available for a device to connect with HealthVault.

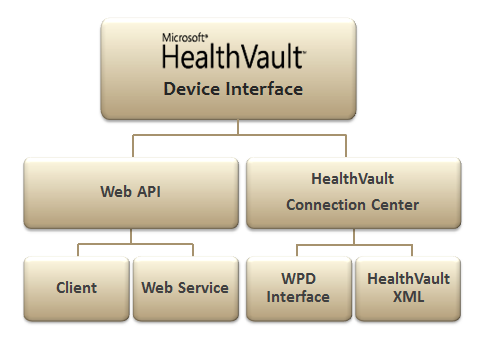


Fig 3-7. Interfaces for Device Integration with HealthVault.

Currently, a large number of devices interface with HealthVault through HealthVault’s Windows client utility, called HealthVault Connection Center. HealthVault Connection Center enables device integration using the Windows Portable Devices (WPD) standard.

If a device already has a Windows device driver, the appropriate data can be communicated to HealthVault using WPD standard. The HealthVault team has a Device Development Kit (DDK) that can be used for this integration. It lies outside the scope of this book.

When there are no WPD supported elements for a device, it can still integrate with HealthVault through HealthVault Connection Center by sending and receiving HealthVault XML directly, Chapter 4 describes the HealthVault data type XML. This approach referred to as HealthVault XML in Fig 3-7. The HealthVault DDK has an example of how to go about configuring such an interface.

In addition to interfacing devices through HealthVault Connection Center, device manufacturers can write their own client application to enable data to upload to HealthVault using the HealthVault Client SDKs. ECG Glove which is available at <http://ineedmd.com/>, is a good example of a device that sends information to HealthVault using this interface.

Devices such as Fitbit and Withings actually take integration a step further and interface with HealthVault directly through the cloud using HealthVault APIs.

### Continua

Continua Health Alliance is a non-profit, open industry organization of healthcare and technology companies joining together in collaboration to improve the quality of personal connected healthcare. With more than 230 member companies around the world, it is the leading consortium for personal healthcare devices. HealthVault has announced support for Continua drivers will be available in the future. In the future, devices will be able to play well in HealthVault and Continua ecosystem either by using the HealthVault Web API or converting data into IEEE 11073 formats.

Continua is not a standards body, but has identified a set of standards that together enable a personal connected healthcare ecosystem. At heart of it is the IEEE 11073 Personal Health Data Standard, which dictates various data standard profiles for devices ranging from blood pressure cuffs to weighing scales. IEEE 11073 is a data standard and is independent of transport.

On the transport layer, Continua supports USB personal Health Class devices, Bluetooth Healthcare device profiles, and other transports as they become complaint in the future.

Figure 3-8 shows the interfaces supported by Continua.

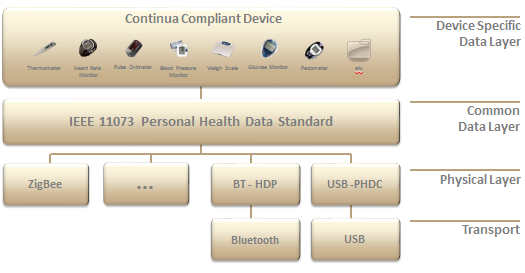


Fig 3-8. Continua complaint devices

## Application Connectivity

As of this writing, there are more than 300 HealthVault applications are live in the United States. HealthVault Applications work with the HealthVault personal health data store by using various APIs over the HTTP protocol, as we have seen with PutThings and GetThings. Fig 3-9 depicts various ways in which applications have interfaced with HealthVault, depending on their use case.

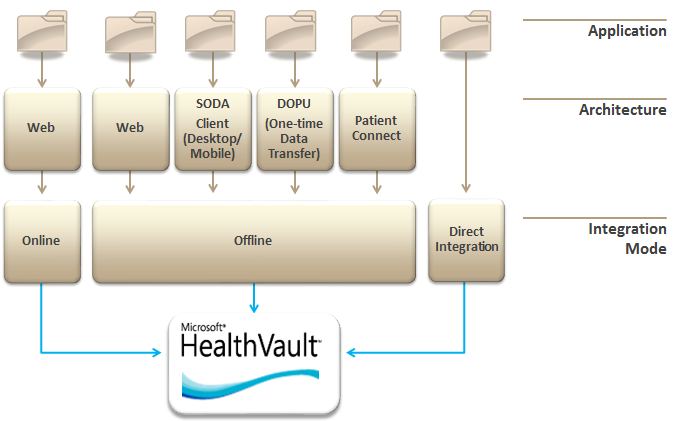


Fig 3-9. Connectivity Types with HealthVault

The following discussion goes into detail on various modes of connecting with HealthVault depending on the application needs as they pertain to the underlying platform, user consent, authentication, and user interface.

### Online HealthVault Application

In addition to data storage, native HealthVault applications can use HealthVault for user authentication and authorization. Any data access using this mode requires the user’s explicit permission each time the application interacts with HealthVault.

Mayo Clinic Health Manager, <https://healthmanager.mayoclinic.com/>, is a Native HealthVault application.

### Offline HealthVault Application

Applications can choose to simply store and access data from HealthVault, without using HealthVault as a primary authentication and authorization entity.

FitBit, <http://www.fitbit.com/>, is a good example of an offline application. It links a FitBit account with a HealthVault user record and account, and then interacts with health items therein.

#### Drop Off Pick Up

Drop Off Pick Up (DOPU) is analogous to sending a secured fax to HealthVault user. The data flow in this architecture is one -way. The application drops the data into HealthVault and the user picks it up.

If a consumer happens to visit a health care institution and does not intend to have an on-going relationship with the care entity, DOPU provides an effective mechanism for the institution to provide documents.

#### Patient Connect

Some entities don’t intend to maintain a public facing web site, but would like to have an ongoing relationship with their users through HealthVault. Patient connect provides an ideal mechanism for such institutions. Via this mechanism, the user authorizes an application to read or write data to their HealthVault record through a user interface on HealthVault.com.

Clinical systems like Electronic Medical Records commonly use this model to connect to HealthVault, which is why this model is called Patient Connect. It should be noted, however, that this model is not limited to clinical systems and can be used by any back-end system.

#### Client Connectivity

Client Connectivity referred as Software on Device Authentication (SODA) enables applications to run on client platforms like desktop or mobile device, outside of the web browser. Every time a user installs a SODA application, the user must authorize that installation of the application to access their HealthVault record. For instance, if the user is running the same application on both their laptop and desktop, they will need to authorize both installations to access their HealthVault record.

A number of mobile health applications like iTriage, LiveScape, and Weight4Me use this architecture. In Chapter 5 we will develop a mobile application and do a detailed walk through of the APIs available to use this type of interface.

### Direct Integration

The Direct project, formerly known as NHIN Direct, is collaboration between the public and private sector to develop a simple, secure, and standards-based method to send encrypted health information directly to known, trusted recipients over the Internet. This project aims at replacing the fax machine in Healthcare. Providers are able to send documents to each other securely.

Direct integration is the easiest kind of integration with HealthVault; trusting applications can actually send and receive documents to and from HealthVault using a Direct-enabled e-mail address. HealthVault users get an e-mail address in the format [handle]@direct.healthvault.com. As part of its Direct implementation, HealthVault automatically adds any recognized attachments like Clinical Care Documents or Clinical Care Records to the user’s record.

Google Health was able to interface with HealthVault using the Direct integration. For HealthVault to accept e-mails from a new direct domain, the application needs to register the public key with HealthVault. If the application e-mails to [newuser@direct.healthvault.com](mailto:newuser@direct.healthvault.com) with the user’s email address in the subject line, HealthVault stores the e-mail in a password encrypted package and send an e-mail to the user to associate the dropped off information to their record. The user can also sign in to their HealthVault record and read the message in HealthVault Message Center.

### Application Provisioning and Master Applications

Application Provisioning refers to providing an application in HealthVault’s production environment, which the HeathVault team does for all of connectivity models discussed so far. However, in special cases it provides the ability for applications known as “master” applications to provision individual HealthVault “child” applications.

Frequently, solution providers develop HealthVault integration for common scenarios such as uploading lab information or sending clinical care record information from a facility’s electronic medical record system. These solutions are deployed separately for each institution. The HealthVault team delegates the responsibility for creating these individual application instances to the solution provider through the Master Application mechanism.

Thus, for instance, if we wanted to deploy an individual instance of a Weight Tracker application per institution, we would use the AddApplication API available from the HealthVault Platform. The Appendix has links to examples and resources about how to create a child application.